

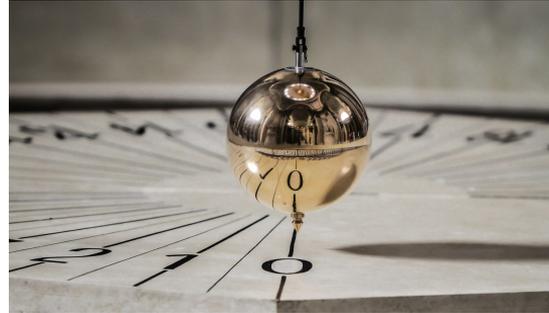


Python Decal Group 8: Simulating a 3D double pendulum!

By Christopher Loewenthal, Cameron Colleran, Izzy Arcoleo, Akhil Korupolu, and Jacqueline Frame

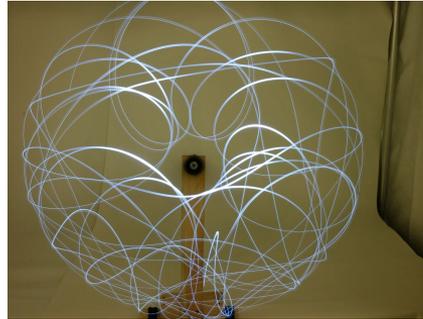
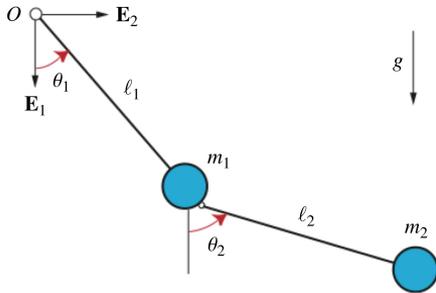
What is a pendulum?

- A pendulum is a weight hung from a fixed point.
 - Some Examples:
 - Playground swing
 - Clocks
 - Seismometers
 - Mechanical Metronomes
 - Spiderman



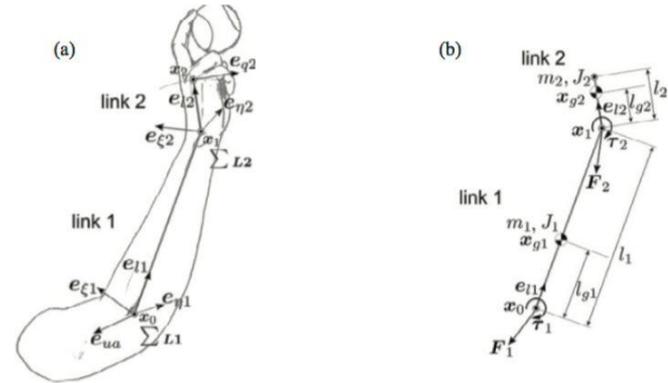
How about a double pendulum?

- Double pendulum is a pendulum that has another pendulum attached at its end.
 - Double pendulums are known as “chaos pendulums”



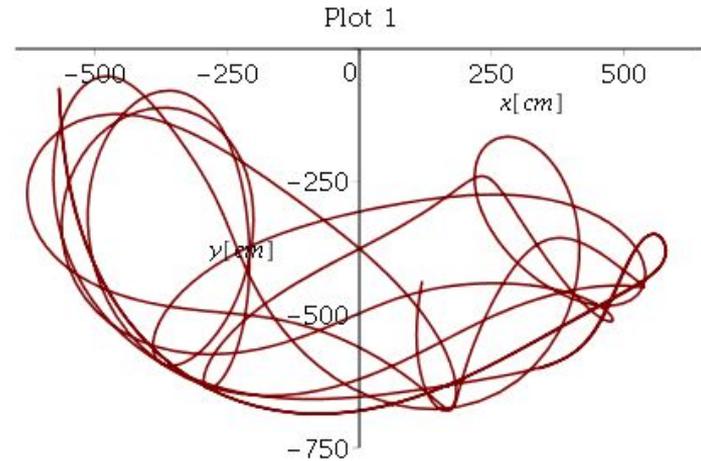
Applications of Double Pendula

- Double pendulums are widely used in education and research
 - Used in studying chaos and state transitions (hence the name chaos pendulum)
 - Study done by Coastal Carolina University uses the double pendulum to analyze and maximize sport performance
 - Discusses how the core movements of golf and baseball can be modeled by double pendulums.



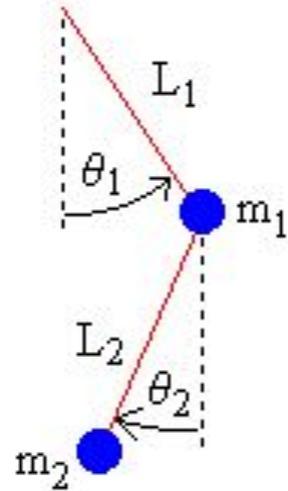
Double Pendula - The Math

- The Double Pendulum is an example of a “chaotic” system
 - Chaotic systems are those that change drastically based off initial conditions and appear to have “random” periodicity
 - Despite the seeming randomness, the behavior is in fact deterministic



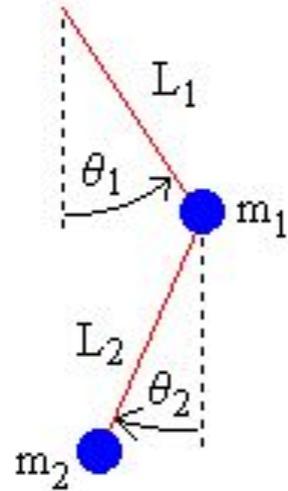
Double Pendula - The Math (cont'd)

- 2D Case
- Kinematics
 - x - horizontal position of mass
 - y - vertical position of mass
 - θ - angle of pendulum
 - L - length of string



Double Pendula - The Math (cont'd)

- Kinematic equations for position of masses:
 - $x_1 = L_1 \sin \theta_1$
 - $y_1 = -L_1 \cos \theta_1$
 - $x_2 = x_1 + L_2 \sin \theta_2 = L_1 \sin \theta_1 + L_2 \sin \theta_2$
 - $y_2 = y_1 - L_2 \cos \theta_2 = -L_1 \cos \theta_1 - L_2 \cos \theta_2$





Double Pendula - The Math (cont'd)

- Determining velocity of masses:

- $x'_1 = \theta'_1 L_1 \cos\theta_1$
- $y'_1 = \theta'_1 L_1 \sin\theta_1$
- $x'_2 = x'_1 + \theta'_2 L_2 \cos\theta_2$
- $y'_2 = y'_1 + \theta'_2 L_2 \sin\theta_2$

- Determining acceleration of masses:

- $x''_1 = -\theta'^2_1 L_1 \sin\theta_1 + \theta''_1 L_1 \cos\theta_1$
- $y''_1 = \theta'^2_1 L_1 \cos\theta_1 + \theta''_1 L_1 \sin\theta_1$
- $x''_2 = x''_1 - \theta'^2_2 L_2 \sin\theta_2 + \theta''_2 L_2 \cos\theta_2$
- $y''_2 = y''_1 + \theta'^2_2 L_2 \cos\theta_2 + \theta''_2 L_2 \sin\theta_2$



Nonlinear system of Lagrange differential equations

$$(m_1 + m_2)L_1\theta''_1 + m_2L_2\theta''_2\cos(\theta_1 - \theta_2) + m_2L_2\theta'^2_2\sin(\theta_1 - \theta_2) + (m_1 + m_2)g\sin(\theta_1) = 0$$

$$L_2\theta''_2 + L_1\theta''_1\cos(\theta_1 - \theta_2) - L_1\theta'^2_1\sin(\theta_1 - \theta_2) + g\sin(\theta_2) = 0$$

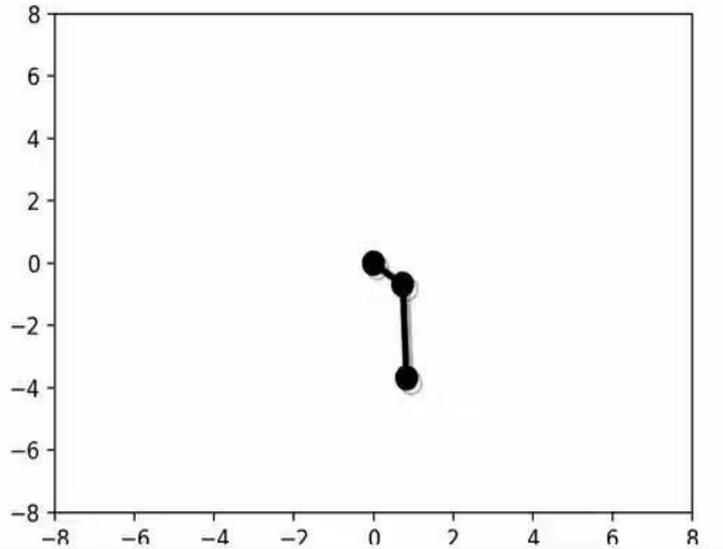


2D Animation of a Double Pendulum

- Used sympy to write equations symbolically to make it easier to find derivatives /manipulate equations and made code clearer
- Wrote equations for kinetic and potential energy to use Lagrange's equations to find equations of motion for the pendulum (we talk about solving more in later slides)
- Then converted symbolic equations to numerical to be able to solve the system of differential equations using odeint
- Used the equations for x and y coordinates to find coordinates - make array of coordinates
- Set points on a plot to be the coordinates of the origin and each pendulum bob using data from array - coordinates change with each frame of animation
- Connect points with lines
- Also added opportunity for user input for masses of pendulum bob's and length of strings to see how that affects the motion



Screen Recording of 2D pendulum animation



Mass 1: 1 kg

Mass 2: 2 kg

Length 1: 1 m

Length 2: 3 m



3D Animation of a Double Pendulum

Packages used: numpy, sympy, scipy, and matplotlib, vpython

Defined variables in sympy and turned azimuthal angles into functions with respect to time. Define equations for kinetic and potential energy to get Lagrange equations.

Used `smp.solve` (which only seems to work in version 1.6.2 of sympy) to solve , yielding four 2nd order ODE's.

Used dummy variables to change 2nd order ODE's into 8 1st order ODE's that the computer can solve more easily.

Used `smp.lambdify` to translate sympy expressions into numerical functions that python can solve.

Defined a function that takes in our 8 equations so that we can solve using `odeint` given initial conditions. Data is in terms of polar coordinates. Used 1001 data points for use in animation.

Defined a function that takes the polar coordinates and changes them into set of cartesian coordinates.

Saved position vectors as a numpy array that can be fed into the animation.

Using Vpython

VPython
3D Programming for Ordinary Mortals

Web VPython: webvpython.org
(formerly GlowScript VPython)

Examples

Documentation

Web VPython User Forum

VPython 7 User Forum

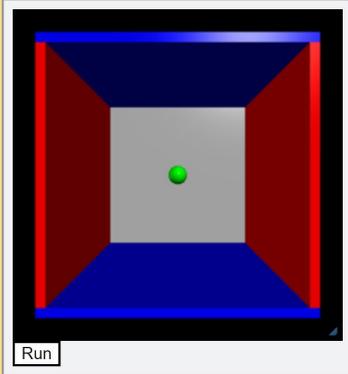


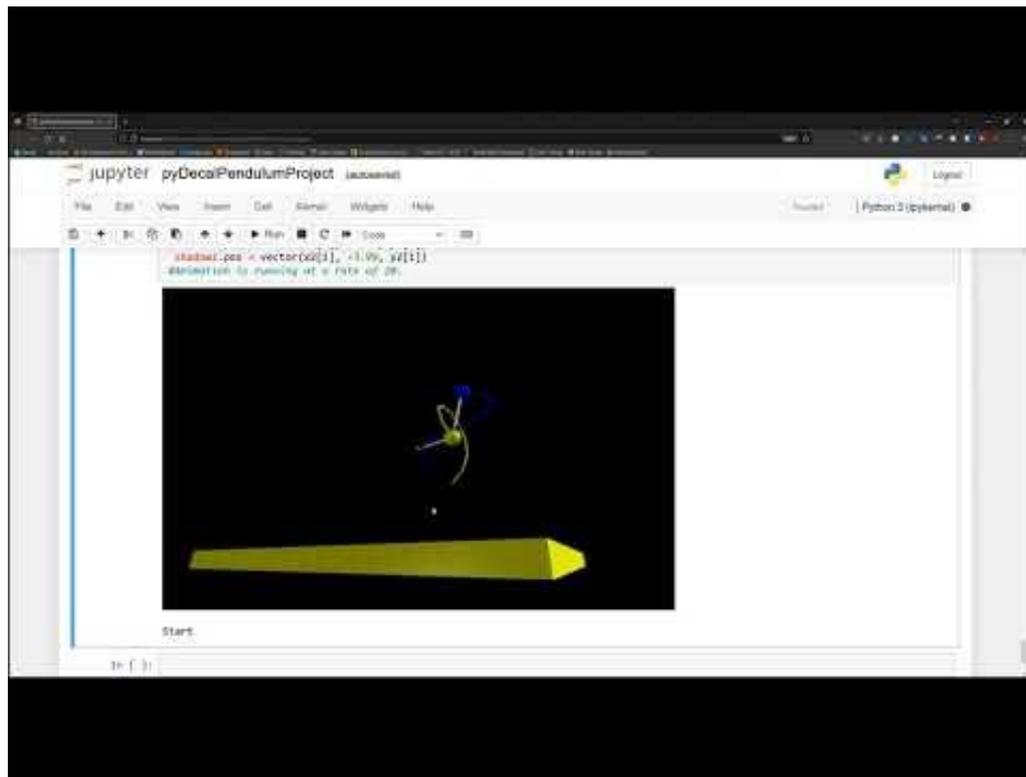
Image from <https://www.vpython.org/>

Vpython is a great introduction to 3D animation for beginners. Useful for making basic mechanical models.

Library consists of basic 3D shapes.

Uses a cartesian coordinate system.

With a touchscreen device, you can view objects from different angles and zoom in or out.





Resources

<https://www.vpython.org/>

<https://www.youtube.com/@MrPSolver>

<https://www.youtube.com/watch?v=MtG9cueB548>

<https://www.youtube.com/@paulmcwhorter>

<https://www.myphysicslab.com/pendulum/double-pendulum-en.html>

<https://digitalcommons.coastal.edu/cgi/viewcontent.cgi?article=1077&context=bridges>

Vpython.org is a great place to get inspiration for projects and see questions asked by other users.

Mr. PSolver has a Youtube channel filled with coding tutorials and walkthroughs to show you how to write code to solve math equations and make animations of physics models.

The third link is the video that I used as inspiration for the 3D animation

Paul McWhorter has a Youtube channel containing videos of basic Vpython tutorials for creating shapes and basic animation.

MyPhysicsLab delves into the physics and math behind the double pendulum

Last link is the research paper written about Sport Performance and Double Pendulums by Coastal Carolina University.